66824-US
TOK/NS

日本国特許庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日
Date of Application: ２００３年　２月１８日

出願番号
Application Number: 特願２００３－０４００５６

[ST. 10/C]: [ＪＰ２００３－０４００５６]

出願人
Applicant(s): 株式会社デンソー

２００３年１２月１９日

特許庁長官
Commissioner,
Japan Patent Office 今井康夫

出証番号　出証特２００３－３１０５５３２

【書類名】　特許願
【整理番号】　PNID4203
【提出日】　平成15年 2月18日
【あて先】　特許庁長官殿
【国際特許分類】　G06F　9/46
【発明者】
　【住所又は居所】　愛知県刈谷市昭和町１丁目１番地　株式会社デンソー内
　【氏名】　西村　忠治
【特許出願人】
　【識別番号】　000004260
　【氏名又は名称】　株式会社デンソー
【代理人】
　【識別番号】　100082500
　【弁理士】
　【氏名又は名称】　足立　勉
　【電話番号】　052-231-7835
【手数料の表示】
　【予納台帳番号】　007102
　【納付金額】　21,000円
【提出物件の目録】
　【物件名】　明細書　1
　【物件名】　図面　1
　【物件名】　要約書　1
　【包括委任状番号】　9004766
【プルーフの要否】　要

【書類名】明細書

【発明の名称】タスク間通信方法、プログラム、記録媒体、電子機器

【特許請求の範囲】

【請求項１】

あるタスク（以下送信側タスクと称する）上での処理から、前記送信側タスクとは別のタスク（以下受信側タスクと称する）上での処理に対するデータの送信要求が発生した場合に、前記送信側タスク上での処理において、当該データを、当該受信側タスク上での処理から取得可能なキューに格納し、当該受信側タスクの起床要求をオペレーティングシステムに対して行い、当該起床要求に基づいて前記オペレーティングシステムの処理によって当該受信側タスクが起床された際に、当該受信側タスク上での処理において、当該受信側タスク上での処理から取得可能な前記キューに格納されたデータを取得する処理を、コンピュータによって実行することによって実現されるタスク間通信方法において、

前記送信側タスク上での処理における前記受信側タスクの起床要求の回数を前記データの送信回数よりも少ない回数とし、

前記受信側タスク上での処理における前記キューに格納されたデータを取得する処理において、前記キューに格納されたデータを複数取得すること

を特徴とするタスク間通信方法。

【請求項２】

あるタスク（以下送信側タスクと称する）上での処理から、前記送信側タスクとは別のタスク（以下受信側タスクと称する）上での処理に対するデータの送信要求が発生した場合に、前記送信側タスク上での処理において、当該データを、当該受信側タスク上での処理から取得可能なキューに格納し、当該受信側タスクの起床要求をオペレーティングシステムに対して行い、当該起床要求に基づいて前記オペレーティングシステムの処理によって当該受信側タスクが起床された際に、当該受信側タスク上での処理において、当該受信側タスク上での処理から取得可能な前記キューに格納されたデータを取得する処理を、コンピュータによって実行することによって実現されるタスク間通信方法において、

前記送信側タスク上での処理で、前記受信側タスク上での処理に対するデータ

の送信要求が発生した場合に、当該受信側タスク上での処理から取得可能な前記キュー内に、前記キューへの当該データの格納以前にすでに格納されているデータがあるか否かを判定し、ある場合には前記送信側タスク上での処理における当該受信側タスクに対する起床要求を行わず、ない場合にのみ前記起床要求を行い、

前記受信側タスクでは、当該受信側タスク上での処理から取得可能な前記キューに存在するすべてのデータを取得すること

を特徴とするタスク間通信方法。

【請求項３】

あるタスク（以下送信側タスクと称する）上での処理から、前記送信側タスクとは別のタスク（以下受信側タスクと称する）上での処理に対するデータの送信要求が発生した場合に、前記送信側タスク上での処理において、当該データを、当該受信側タスク上での処理から取得可能なキューに格納し、当該受信側タスクの起床要求をオペレーティングシステムに対して行い、当該起床要求に基づいて前記オペレーティングシステムの処理によって当該受信側タスクが起床された際に、当該受信側タスク上での処理において、当該受信側タスク上での処理から取得可能な前記キューに格納されたデータを取得する処理を、コンピュータによって実行することによって実現されるタスク間通信方法において、

前記送信側タスク上での処理で、前記受信側タスク上での処理に対するデータの送信要求が発生した場合に、当該受信側タスクに対する前記起床要求が現在あるか否かを判定し、ある場合には前記送信側タスク上での処理における当該受信側タスクに対する起床要求を行わず、ない場合にのみ前記起床要求を行い、

前記受信側タスクでは、当該受信側タスク上での処理から取得可能な前記キューに存在するすべてのデータを取得すること

を特徴とするタスク間通信方法。

【請求項４】

請求項１～３のいずれかに記載のタスク間通信方法において、

前記オペレーティングシステムは、前記起床要求があって未だ実行していないタスクを特定するためのタスク特定情報を記憶し、記憶されたタスク特定情報に

対応する優先度が一番高いタスクから順に、タスクを起床させるものであり、

すべてのタスクを異なる優先度に設定し、前記キューは、当該優先度ごとに設けること

を特徴とするタスク間通信方法。

【請求項５】

請求項１～４のいずれかに記載のタスク間通信方法の各処理をコンピュータに実行させるためのプログラム。

【請求項６】

請求項５に記載のプログラムを記録した記録媒体。

【請求項７】

請求項５に記載のプログラムと、当該プログラムを実行するコンピュータとを備えることを特徴とする電子機器。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

タスク間通信方法等に関する。

【０００２】

【従来の技術】

従来からリアルタイムオペレーティングシステム（ＲＴＯＳ）等によって管理されるタスク間で、データを含むメッセージの通信（送受信、やりとり、受け渡し）を行うための種々のタスク間通信方法が知られている（例えば、特許文献１参照。）。

【０００３】

例えば、図４（ａ）は、メッセージを送信する側のタスク内（あるいはそのタスクから呼び出すサブルーチン内）におけるメッセージ送信処理の流れを示すフローチャートである。また図４（ｂ）は、メッセージ内のデータを受信（取得）する側のタスク（あるいはそのタスクから呼び出すサブルーチン内）におけるデータ受信利用処理の流れを示すフローチャートである。

【０００４】

ここで、メッセージの送信側タスクをタスクＡ、メッセージに基づいてＲＴＯＳによって起床されメッセージ内のデータを受信して利用する受信側タスクをタスクＢとして説明する。タスクＡは優先度Ａであり、タスクＢは優先度Ｂであり、優先度Ａは優先度Ｂよりも優先度が高いものとする。ＲＴＯＳは、各タスクから起床待ちキュー（Ｒｅａｄｙ　Ｑｕｅｕｅ）にキューイングされたタスクのうち、優先度の最も高いタスクを起床させるスケジューリング方法を採るものとする。

【０００５】

例えば、タスクＡ内でイベントの発生を検知した場合に、そのイベントに関連するデータを、タスクＢのそのイベント用の処理ルーチンで処理させるためにメッセージ通信処理が行われる。

タスクＡにおけるメッセージ送信処理は、図４（ａ）に示すように、送信対象のデータをメッセージ送信先のタスク用のキュー、すなわち、タスクＢ用のキュー（Ｑｕｅｕｅ）にキューイング（保存）し（Ｓ１１）、ＲＴＯＳに対し、対応する送信先タスク（受信側タスク）、すなわちタスクＢの起床要求を行う（Ｓ１２）処理である。

【０００６】

ＲＴＯＳは、この起床要求を受け、この起床要求されたタスクを特定するための識別情報（タスクＩＤ）をタスクの起床待ちキューにキューイングする。そして、優先度Ａのタスクの処理が完了すると、ＲＴＯＳはタスクＢの優先度よりも高い優先度のタスクのタスクＩＤが起床待ちキューになくなった時点で、起床待ちキューにキューイングされているタスクＩＤに基づいてタスクＢを起床し、タスクＢを実行する。

【０００７】

このタスクＢ内では、図４（ｂ）に示す処理が行われる。つまり、自タスク用のキュー、すなわちタスクＢのキューからデータを取り出し（Ｓ２１）、そのデータを用いた処理を行う（Ｓ２２）。このようにして、タスクＡ内の処理からタスクＢを起床させてタスクＢ内の処理を実行させ、このタスクＢ内の処理で、タスクＡ内の処理によってキューイングされたデータを受け取って利用することが

できる。例えば、タスクＡ内で検知したイベントの処理に必要なデータを、タスクＢ内のこのイベントの処理ルーチンに渡して処理させることができる。

【０００８】

【特許文献１】

特開２００２－１８９６０６号公報

【０００９】

【発明が解決しようとする課題】

こうした従来のタスク間通信方法では、優先度が高いタスクＡ内で、優先度の低いタスクＢに対してメッセージを一度に複数回送信する場合、図４（ａ）に示したメッセージ送信処理は、複数回実行されることとなる。

【００１０】

したがって、メッセージを送信する回数分のデータが、一旦、送信先のタスクＢ用のキューにキューイングされるとともに、タスクＢの起床要求がＲＴＯＳのタスクの起床待ちキュー（Ｒｅａｄｙ　Ｑｕｅｕｅ）にキューイングされる。

そして、その後、タスクＢよりも優先度の高い処理（タスク、割り込み）が無くなった時点で、１．ＲＴＯＳによるタスクＢの起床処理　→　２．タスクＢ内でのデータ受信処理（Ｓ２１）　→　３．タスクＢにおける受信したデータを用いた処理（Ｓ２２）　→　４．タスクＢ終了という一連のメッセージ配送処理を、その複数回分繰り返し実行する必要がある。

【００１１】

例えば、図５は、図４（ａ）の処理を行うルーチンをSendMessage関数、図４（ｂ）のＳ２１の処理を行うルーチンをReceiveMessage関数としてサブルーチン化して持つイベントサービスを介して、タスクＡからタスクＢ内のイベントの処理ルーチンにデータＡ，データＢ，データＣに基づく処理を実行させるための３つのメッセージを送る場合のタスク間メッセージ通信の処理の流れを示す説明図である。なお、イベントサービスはＲＴＯＳの一部として構成することもでき、タスクの一部（サブルーチン）として構成することもできる。ここでは、イベントサービスによる処理は、イベントサービスの関数の呼び出し元と同じ実行レベル（優先度）で実行されるものとして説明する。

【００１２】

図５に示すように、タスクＡ内でイベントサービスのSendMessage関数を呼び出し、データＡをその引数としてイベントサービスへ渡す。イベントサービスは、この関数内でタスクＢ用のキューにデータＡをキューイングし（図４（ａ）のＳ１１に相当する）、ＲＴＯＳに対してタスクＢの起床要求をする（図４（ａ）のＳ１２に相当する）。そしてこの起床要求に基づきＲＴＯＳは、起床待ちキューにタスクＢのタスクＩＤをキューイングする。そして、再びタスクＡは、SendMessage関数を呼ぶことでデータＢをイベントサービスへ渡す。イベントサービスはタスクＢ用のキューにデータＢをキューイングし（Ｓ１１）、ＲＴＯＳに対してタスクＢの起床要求をする（Ｓ１２）。この要求に基づきＲＴＯＳは、起床待ちキューにタスクＢのタスクＩＤをキューイングする。そして、再びタスクＡは、SendMessage関数を呼ぶことでデータＣをイベントサービスへ渡す。イベントサービスはタスクＢのキューにデータＣをキューイングし（Ｓ１１）、ＲＴＯＳに対してタスクＢの起床要求をする（Ｓ１２）。この要求に基づきＲＴＯＳは、起床待ちキューにタスクＢのタスクＩＤをキューイングする。そして、タスクＡのすべての処理が終了すると、ＲＴＯＳへ処理が移行する。

【００１３】

ＲＴＯＳは、起床待ちキューを参照して、このキュー内で優先度の最も高いタスクＩＤに対応するタスクを起床させる。起床待ちキューには、タスクＢのタスクＩＤがキューイングされているので、タスクＢを起床させる（図５における一番上のSchedule)。ここで起床待ちキューに残されたタスクＢのタスクＩＤは残り２つとなる。起床されたタスクＢ内の処理では、イベントサービスのReceiveMessage関数を呼ぶ。イベントサービスは、ReceiveMessage関数の処理として、タスクＢ用にキューイングされているデータＡを取り出し（図４のＳ２１に相当する）、タスクＢにそのデータＡを返す。そしてタスクＢは、イベントサービスから受け取ったデータＡを用いた処理を実行する（図４のＳ２２に相当する）。そしてタスクＢのその他の処理が終了すると、処理をＲＴＯＳへ移行させる。ＲＴＯＳは、起床待ちキューを参照して、優先度の高い順にタスクを起床させる。起床待ちキューには、タスクＢがキューイングされているので、タスクＢを起床さ

せる。ここで起床待ちキューに残されたタスクＢのタスクＩＤは残り１つとなる。起床されたタスクＢ内の処理では、イベントサービスのReceiveMessage関数を呼ぶ。イベントサービスは、ReceiveMessage関数の処理として、タスクＢ用にキューイングされているデータＢを取り出し（Ｓ２１）、タスクＢにそのデータＢを返す。そしてタスクＢは、イベントサービスから受け取ったデータＢを用いた処理を実行する（Ｓ２２）。そしてタスクＢのその他の処理が終了すると、処理をＲＴＯＳへ移行させる。ＲＴＯＳは、起床待ちキューを参照して、優先度の高い順にタスクを起床させる。起床待ちキューにはまだタスクＢがキューイングされているので、タスクＢを起床させる。ここで起床待ちキューは空になる。起床されたタスクＢ内の処理では、イベントサービスのReceiveMessage関数を呼ぶ。イベントサービスは、ReceiveMessage関数の処理として、タスクＢ用にキューイングされているデータＣを取り出し（Ｓ２１）、タスクＢにそのデータＣを返す。そしてタスクＢは、イベントサービスから受け取ったデータＣを用いた処理を実行する（Ｓ２２）。そしてタスクＢのその他の処理が終了すると、処理をＲＴＯＳへ移行させる。

【００１４】

このように、送信側タスク（タスクＡ）内で処理をＲＴＯＳへ返すことなく、送信先のタスク（タスクＢ）にメッセージを複数回送信した場合に、受信側タスクであるタスクＢは、単にタスクＢ用のキュー内のデータを受信してそのデータを用いる処理を順次実行するだけにもかかわらず、１メッセージ毎にタスクの起床・終了処理（ＲＴＯＳにおける処理（例えば図５のscheduleのための処理など）やタスク内での処理（例えば図４（ｂ）の処理へ移行するための判定処理など）が発生し、処理装置（例えばＣＰＵ）の処理負荷が大きくなるという問題がある。

【００１５】

そこで本願発明は、処理装置の処理負荷を低減させることのできるタスク間通信方法等を提供することを目的とする。

【００１６】

【課題を解決するための手段及び発明の効果】

上述した問題点を解決するためになされた請求項１に記載のタスク間通信方法は、送信側タスク上での処理から受信側タスク上での処理に対するデータの送信要求が発生した場合に、送信側タスク上での処理において、そのデータを、受信側タスク上での処理から取得可能なキューに格納し、その受信側タスクの起床要求をオペレーティングシステムに対して行う。このとき、送信側タスク上での処理における前記受信側タスクの起床要求の回数をデータの送信回数よりも少ない回数とする。そして、その起床要求に基づいてオペレーティングシステムの処理によって受信側タスクが起床された際に、その受信側タスク上での処理において、その受信側タスク上での処理から取得可能な前記キューに格納されたデータを複数取得する処理を行う。

【００１７】

したがって、送信側タスクでは複数のデータの送信要求に対して、１の起床要求を行い、その起床要求によって起床された受信側タスクでは、複数のデータをキューから取得するため、従来のように１つのデータ（例えばメッセージ）を送信する毎に、受信側タスクの起床・終了処理が行われてしまい、処理装置の処理負荷が大きくなるという問題を抑えることができ、処理装置の処理負荷を低減させることができる。

【００１８】

また、上述した問題点を解決するためになされた請求項２に記載のタスク間通信方法は、送信側タスク上での処理から、受信側タスク上での処理に対するデータの送信要求が発生した場合に、送信側タスク上での処理において、そのデータを、その受信側タスク上での処理から取得可能なキューに格納し、そのキュー内に、前記キューへのそのデータの格納以前にすでに格納されているデータがあるか否かを判定し、ある場合にはその受信側タスクに対する起床要求を行わず、ない場合にのみ前記起床要求を行う。そして、起床要求に基づいて前記オペレーティングシステムの処理によってその受信側タスクが起床された際に、その受信側タスク上での処理において、その受信側タスク上での処理から取得可能な前記キューに存在するすべてのデータを取得する。

【００１９】

このようにすれば、送信側タスク上の処理で、連続してデータ（例えばメッセージ）を受信側タスク上の処理へ送る場合に、その受信側タスクの起床要求は１回のみで済む。したがって、従来のように１つのデータ（例えばメッセージ）を送信する毎に、受信側タスクの起床・終了処理が行われてしまい、処理装置の処理負荷が大きくなるという問題を抑えることができ、処理装置の処理負荷を低減させることができる。

【００２０】

また、上述した問題点を解決するためになされた請求項３に記載のタスク間通信方法は、送信側タスク上での処理から、受信側タスク上での処理に対するデータの送信要求が発生した場合に、送信側タスク上での処理において、そのデータを、その受信側タスク上での処理から取得可能なキューに格納し、その受信側タスクに対する起床要求が現在あるか否かを判定し、ある場合にはその受信側タスクに対する起床要求を行わず、ない場合にのみ起床要求を行う。そして、起床要求に基づいて前記オペレーティングシステムの処理によってその受信側タスクが起床された際に、その受信側タスク上での処理において、その受信側タスク上での処理から取得可能な前記キューに存在するすべてのデータを取得する。

【００２１】

このようにすれば、送信側タスク上の処理で、連続してデータ（例えばメッセージ）を受信側タスク上の処理へ送る場合に、その受信側タスクの起床要求は１回のみで済む。したがって、従来のように１つのデータ（例えばメッセージ）を送信する毎に、受信側タスクの起床・終了処理が行われてしまい、処理装置の処理負荷が大きくなるという問題を抑えることができ、処理装置の処理負荷を低減させることができる。

【００２２】

特に、オペレーティングシステムがタスクの優先度に基づいて優先度の高いタスクから順に実行する構成の場合、請求項４に示すように、すべてのタスクを異なる優先度に設定し、前記キューはその優先度ごとに設けるようにするとよい。このようにすれば、優先度に基づいてタスクやキューを管理できる。すなわち、請求項１～３に記載の「受信側タスク上での処理から取得可能なキュー」は、優

先度ごとに設けたキューとすることができる。例えば、データ（メッセージ）送信時に、他に送信待ちの同一優先度のメッセージがキューに有るか否かをチェックし、有る場合にはオペレーティングシステムに対するその優先度のタスクの起床要求を行わず、ない場合のみその優先度のタスクの起床要求を行うようにできる。なお、このタスクの起床要求がすでにあるか否かは、例えば請求項２に示すようにその優先度のキューにデータがすでにあるか否かに基づいて判定してもよいし、請求項３に示すようにその受信側タスクに対する前記起床要求が現在あるか否かに基づいて判定してもよい。そして、受信側タスク上での処理では、データの受信処理時に、その優先度のキューにメッセージがあれば、そのメッセージをすべて取得するように構成できる。このように構成することで、例えば、送信側タスクで同一優先度のメッセージが複数連続して送信要求され、かつ、受信側タスクが送信側タスクに比べ優先度が同じか低い場合に、受信側タスクの起床及び終了処理を減らすことができ、処理装置の処理負荷が低減できる。

【００２３】

そして、請求項５に示すように、上述したタスク間通信方法の各処理をコンピュータに実行させるためのプログラムとして構成することができる。このようなプログラムは、例えば、請求項６に示すようにして、フレキシブルディスク、光磁気ディスク、ＣＤ－ＲＯＭ、ハードディスク、ＲＯＭ、ＲＡＭ等のコンピュータ読み取り可能な記録媒体に記録し、必要に応じてコンピュータシステムにロードして起動することにより実行することができる。また、プログラムは、ネットワークを介してロードして起動することにより実行することもできる。また、請求項７に示すように、請求項５に記載のプログラムと、このプログラムを実行するコンピュータを備える電子機器として構成することができる。このような電子機器は、コンピュータ（処理装置）によるタスクの起動・終了処理を減らすことができるため、機器本来の機能を実現するための処理に処理装置の処理時間をより多く割くことが可能となる。

【００２４】

【発明の実施の形態】

以下、本発明を具体化した一実施例について図面を参照して説明する。図１は

、本発明の電子機器としてのエンジン制御装置（以下「ＥＣＵ」という。）１の構成を表すブロック図である。ＥＣＵ１は、車両に搭載された内燃機関型エンジンの制御を行う。

【００２５】

ＥＣＵ１は、エンジンのクランク軸が所定角度回転する毎にパルス状の信号を出力する回転角センサ、エンジンの特定の気筒のピストンが所定位置（例えば上死点：ＴＤＣ）にくる度にパルス状の信号を出力する基準位置センサ、エンジンの冷却水の温度を検出する水温センサ、及び酸素濃度を計測する酸素濃度センサ等、エンジンの運転状態を検出する様々なセンサ３０からの信号を入力して波形整形やＡ／Ｄ変換を行ったり、車内ＬＡＮ３２から信号を入力したりする入力回路２１と、入力回路２１からの入力信号に基づき、エンジンを制御するための様々な処理や車内ＬＡＮ３２を介して他のＥＣＵからデータを受信したり他のＥＣＵに対して送信するデータを生成して送信したりするための様々な処理を実行するマイコン１０と、マイコン１０からのデータに応じて、エンジンに取付けられたインジェクタ（燃料噴射装置）及びイグナイタ（点火装置）等のアクチュエータ４０を駆動したり、車内ＬＡＮ３２にデータを出力したりするための出力回路２２とを備えている。

【００２６】

そして、マイコン１０には、プログラムを実行する周知のＣＰＵ１１と、ＣＰＵ１１によって実行されるプログラムを記憶するＲＯＭ１２と、ＣＰＵ１１による演算結果等を記憶するためのＲＡＭ１３と、入力回路２１及び出力回路２２との間で信号をやり取りするためのＩ／Ｏ１４と、各種レジスタやフリーランカウンタ等（図示省略）とを備えている。

【００２７】

このように構成されたＥＣＵ１は、入力回路２１及び出力回路２２を制御し車内ＬＡＮ３２を介して他のＥＣＵ等と通信を行う通信処理や、各種センサ３０及び車内ＬＡＮ３２から入力回路２１を介して入力される信号に基づき、出力回路２２に接続されたアクチュエータ４０を駆動するエンジン制御処理を行う。

【００２８】

次に、ＲＯＭ１２に記憶される「プログラム」としてのエンジン制御プログラムの構成を説明する。なお、本明細書では、例えば「タスクが・・・する」「ＲＴＯＳが・・・する」「イベントサービスが・・・する」「・・・関数が・・・する」というプログラムを主体とした表現を適宜用いているが、詳しくは、ＣＰＵ１１がＲＯＭ１２に記憶されたこれらのプログラムを実行して処理を行うことは言うまでもない。エンジン制御プログラムは、ＲＴＯＳ、タスクＡ，タスクＢ，タスクｃ…等の各々のタスク、及び、各々のタスクのメッセージ送信処理のサブルーチンであるSendMessage関数とデータ受信処理のサブルーチンであるReceiveMessage関数を備えるイベントサービスを構成するプログラムを備える。各タスクはそれぞれ異なる優先度に設定されており、イベントサービスの各関数（ルーチン）は呼び出し元のタスクと同じ優先度で実行される。

【００２９】

ＲＡＭ１３上には各タスク用のキューのための記憶領域を確保している。キューは、先入れ先出し（ＦＩＦＯ）バッファ（記憶領域）である。

例えば、タスクＡ内の処理においてタスクＢ内の所定のルーチンで処理すべきイベントが発生した場合のように、タスクＡ内の処理からタスクＢ内の処理へデータを送る必要が発生した場合、タスクＡ内の処理で、送るべきデータを引数としてイベントサービスのSendMessage関数を呼ぶ。

【００３０】

このイベントサービスのSendMessage関数による処理を図２（ａ）に示す。まず、SendMessage関数による処理では、引数として受け取った（例えばスタックを介して受け取った）送信対象のデータを、送信先のタスク用のキュー、すなわちタスクＢ用のキューへ、キューイングする（Ｓ１１０）。

【００３１】

そして、この送信先のタスク用のキューのメッセージ数カウンタの値、すなわちタスクＢ用のキューのメッセージ数カウンタの値を、インクリメントする（Ｓ１２０）。なお、各タスク用のメッセージ数カウンタの値はＲＴＯＳの初期化処理で０に設定されている。

【００３２】

次に、この送信先のタスク用のメッセージ数カウンタの値、すなわちタスクＢ用のメッセージ数カウンタの値が、１であるか否かを判定する（Ｓ１３０）。１の場合には（Ｓ１３０：ＹＥＳ）、Ｓ１４０へ移行する。Ｓ１４０では、ＯＳに対して、送信先タスクの起床要求、すなわち、タスクＢの起床要求を行う。ＲＴＯＳは、この起床要求を受け、この起床要求されたタスクを特定するための識別情報（タスクＩＤ）をタスクの起床待ちキューにキューイングする。すなわち、タスクＢのタスクＩＤを起床待ちキューにキューイングする。そして、このSend Message関数による処理を終了し、タスクＡの処理へ復帰する。一方、Ｓ１３０の判定で、メッセージカウンタの値が１以外の場合には（Ｓ１３０：ＮＯ）、Ｓ１４０の処理は行わずに、そのまま、このSendMessage関数による処理を終了し、タスクＡの処理へ復帰する。

【００３３】

このように、送信先のタスク用のメッセージカウンタの値が１の場合、すなわち、送信先のタスク用のキューに最初に送信対象のデータを保存したときにだけ、その送信先のタスクの起床要求をＯＳに対して行い、それ以後、キューに送信対象のデータを追加する際には、ＯＳに対する送信先タスクの起床要求を行わない。

【００３４】

イベントサービスのSendMessage関数から復帰した後、タスクＡはその他のタスク内の処理を行う。そして、タスクＡの処理が終了するとタスクＡはＲＴＯＳへ処理の終了を伝え（処理をＲＴＯＳへ移行して）、終了する。

ＲＴＯＳは、起床待ちキューに保存されたタスクＩＤのうち、もっとも優先度の高いタスクＩＤのタスクを起床させる。すなわちそのタスクへ処理を移行させる。ここでは、データの受信側タスク、すなわちタスクＢが起床された場合について説明する。

【００３５】

このタスクＢ内（データ受信タスク内）では、図２（ｂ）に示すデータ受信利用処理を行う。まず、自タスク用のキュー、つまりタスクＢ用のキューからデータを取り出す（Ｓ２１０）。このＳ２１０の処理はイベントサービスのReceiveM

essage関数をタスクＢ内から呼び出し、イベントサービスのReceiveMessage関数の処理で、タスクＢ用のキューからデータを取り出してタスクＢへ渡すことで行う。そして、タスクＢでは、このデータを用いた処理を行う（Ｓ２２０）。続いて、このタスク用のメッセージカウンタ、すなわちタスクＢ用のメッセージカウンタの値をデクリメントし（Ｓ２３０）、自タスク用のメッセージカウンタの値、すなわちタスクＢ用のメッセージカウンタの値が０であるか否かを判定する（Ｓ２４０）。０でない場合には（Ｓ２３０：ＮＯ）、Ｓ２１０へ移行して、再び、Ｓ２１０～Ｓ２４０の処理を行う。一方、０の場合には、このデータ受信利用処理を終了する。そしてタスクＢのその他の処理を行い、すべての処理が終了すると、タスクＢはＲＴＯＳへ処理の終了を伝え（処理をＲＴＯＳへ移行して）、終了する。

【００３６】

このようにして、送信側タスク内の処理（タスクＡ内の処理）から受信側タスク（タスクＢ）を起床させて、受信側タスク内の処理（タスクＢ内の処理）を実行させ、この受信側タスク内の処理で、送信側タスク内の処理（タスクＡ内の処理）によってキューイングされたデータを受け取って利用することができる。例えば、タスクＡ内で検知したイベントの処理に必要なデータを、タスクＢ内のこのイベントの処理ルーチンに渡して処理させることができる。

【００３７】

ここで、タスクＡからタスクＢへメッセージを３回連続して発行（送信）する場合の処理の例を図３に示して説明する。

図３に示すように、タスクＡ内でイベントサービスのSendMessage関数を呼び出し、データＡをその引数としてイベントサービスへ渡す。イベントサービスは、この関数内でタスクＢ用のキューにデータＡをキューイングし（図２（ａ）のＳ１１０に相当する）、タスクＢ用のメッセージ数カウンタの値をインクリメントする（Ｓ１２０）。その結果、メッセージ数カウンタの値は１となるので（Ｓ１３０：ＹＥＳ）、ＲＴＯＳに対してタスクＢの起床要求を行う（Ｓ１４０）。そしてこの起床要求に基づきＲＴＯＳは、起床待ちキューにタスクＢのタスクＩＤをキューイングする。そして、再びタスクＡは、SendMessage関数を呼ぶこと

でデータＢをイベントサービスへ渡す。イベントサービスはタスクＢ用のキューにデータＢをキューイングし（Ｓ１１０）、タスクＢ用のメッセージカウンタの値をインクリメントする。その結果、メッセージ数カウンタの値は２となるので、そのままタスクＡの処理へ、復帰する（Ｓ１３０：ＮＯ）。すなわち図３に示すように、２つ目のデータであるデータＢをキューに入れた際には、タスクＢの起床要求は行わない。そして、再びタスクＡは、SendMessage関数を呼ぶことでデータＣをイベントサービスへ渡す。イベントサービスはタスクＢのキューにデータＣをキューイングし（Ｓ１１０）、タスクＢ用のメッセージカウンタの値をインクリメントする。その結果、メッセージ数カウンタの値は３となるので、そのままタスクＡの処理へ、復帰する（Ｓ１３０：ＮＯ）。すなわち図３に示すように、３つ目のデータであるデータＣをキューに入れた際には、タスクＢの起床要求は行わない。そして、タスクＡのすべての処理が終了すると、ＲＴＯＳへ処理を移行する。

【００３８】

ＲＴＯＳは、起床待ちキューを参照して、このキュー内で優先度の最も高いタスクＩＤに対応するタスクを起床させる。起床待ちキューには、タスクＢのタスクＩＤがキューイングされているので、タスクＢを起床させる（図３におけるSchedule）。ここで起床待ちキューは空になる。起床されたタスクＢ内の処理では、イベントサービスのReceiveMessage関数を呼ぶ。イベントサービスは、ReceiveMessage関数の処理として、タスクＢ用にキューイングされているデータＡを取り出し（図４のＳ２１０に相当する）、タスクＢにそのデータＡを返す。そしてタスクＢは、イベントサービスから受け取ったデータＡを用いた処理を実行する（Ｓ２２０）。そして、タスクＢ用のメッセージ数カウンタの値をデクリメントする（Ｓ２３０）。その結果、タスクＢ用のメッセージ数カウンタの値は、２となるので、Ｓ２４０の判定処理で再びＳ２１０の処理へ移行することとなる。再び、イベントサービスのReceiveMessage関数を呼び、イベントサービスは、ReceiveMessage関数の処理として、タスクＢ用にキューイングされているデータＢを取り出して（Ｓ２１０）、タスクＢにそのデータＢを返す。そしてタスクＢは、イベントサービスから受け取ったデータＢを用いた処理を実行する（Ｓ２２０）

。そして、タスクＢ用のメッセージ数カウンタの値をデクリメントする（Ｓ２３０）。その結果、タスクＢ用のメッセージ数カウンタの値は、１となるので、Ｓ２４０の判定処理で再びＳ２１０の処理へ移行することとなる。再び、イベントサービスのReceiveMessage関数を呼び、イベントサービスは、ReceiveMessage関数の処理として、タスクＢ用にキューイングされているデータＣを取り出して（Ｓ２１０）、タスクＢにそのデータＣを返す。タスクＢは、イベントサービスから受け取ったデータＣを用いた処理を実行する（Ｓ２２０）。そして、タスクＢ用のメッセージ数カウンタの値をデクリメントする（Ｓ２３０）。その結果、タスクＢ用のメッセージ数カウンタの値は、０となるので、このデータ受信利用処理を抜け（Ｓ２４０：ＹＥＳ）、タスクＢのその他の処理へ移行する。そして、タスクＢのすべての処理が終了すると、処理をＲＴＯＳへ返す。

【００３９】

従来は、図５に示したように、タスクＢの起動終了が、送信したメッセージの数（３回）分行われるため、非常にオーバーヘッドが大きかったのに対し、本実施例のタスク間通信方法では、図３に示したように、タスクＢの起動終了は１回のみであり、タスクの起床・終了処理にかかるオーバーヘッドが少なくなる。そのため、ＣＰＵ１１の処理負荷を軽減することができる。

【００４０】

特に、本実施例のようなエンジン制御では、例えば、急激に要求トルクが増加したことをセンサ３０等から入力された信号に基づいて検知し燃料噴射を追加する為の非同期噴射要求、センサ３０等の信号などの異常を検出しダイアグ用メモリに記憶させる為のダイアグ処理要求、外部ノードからの（CANなどによる）車内ＬＡＮ３２を介した通信データの着信に対応する通信処理要求などを、メッセージとしてタスク間で通信する。こうした要求を検出する処理（送信側、すなわちタスクＡ）とは、別の優先度で、この対応をする処理（受信側、すなわち、タスクＢ）を実行したい場合や、複数の異なる優先度での送信処理に対し一つの受信処理を行う場合の受信処理側リソースの排他処理を簡単にする為に、上述したメッセージとしてタスク間通信でデータをやりとりする。

【００４１】

これらのメッセージは、それぞれ要求が検出されるたびに送信され、以下のような場合に複数の受信側タスクが待ち状態になる。

・複数の送信処理が１つの高い優先度のタスク上で順次処理され、それぞれの送信条件が成立した場合。（例えば、４ｍｓ毎に非同期噴射判定と異常検出処理が行われている場合）

・優先度の高い複数のタスクが連続して実行されている時に、それぞれのタスク上の送信処理の送信条件が成立した場合。（例えば、CAN受信割り込みと４ｍｓ毎の非同期噴射判定が重なった場合）

・一つの送信処理から（一つの送信条件により）、連続して複数のメッセージが送信された場合。（例えば、４ｍｓ毎の異常検出処理で、複数のセンサの異常が検出された場合）

このような場合に、上述した処理によって、ＣＰＵ１１の処理負荷を軽減することができるため、エンジンの高回転時の制御性の悪化を防ぐことができる。

【００４２】

なお、本実施例では、タスクＡ内の処理からタスクＢ内の処理へデータを送るものとして説明したが、例えば、タスクＡ内の処理で送り先のタスクを指定するようにしてもよい。例えば、SendMessage関数の引数として、送り先のタスクを指定するようにしてもよい。送り先のタスクの指定は、例えば、タスクＩＤを引数とすることで行うことができる。タスクＩＤで行う場合には、図２（ａ）に示したＳ１１０～１４０の処理において、「タスクＢ用のキュー」という部分を、「引数として受け取ったタスクＩＤに対応するタスク用のキュー」とすることで実現できる。また、例えば、すべてのタスクの優先度が異なるものとして一意に決定されている場合には、送り先のタスクの指定は、この優先度を引数とすることで行うことができる。この場合、タスク別ではなく優先度別にキューを設け、図２（ａ）に示したＳ１１０～１４０の処理において、「タスクＢ用のキュー」という部分を、「引数として受け取った優先度に対応するキュー」とすることで実現できる。すなわち、メッセージ送信時に送信データのキューイングを行った後、該当する優先度のメッセージ送信カウンタの値に基づき、現在送信待ちの同優先度のメッセージが有るか否かを確認する。待ちメッセージが無い場合は従来

方式と同様にタスクの起床要求をＲＴＯＳに対して行うが、待ちメッセージがある場合はタスクの起床要求は行わず送信処理を終了する。そしてＲＴＯＳによって受信側タスクが起床されると、このキューイングしたデータを取り出して、取り出したデータを利用する処理を行った後、さらに同優先度の送信待ちメッセージが残っているかどうかをチェックし、もし残っていればタスクを終了せずに続けて次の受信処理を実施し、送信待ちメッセージがなくなるまで繰り返す。このように構成することで、同一優先度のメッセージが複数連続して送信要求され、かつ、受信側タスクが送信側タスクに比べ優先度が同じか低い場合に、タスクの起床及び終了処理が少なく済み、ＣＰＵ１１の処理負荷が低減できる。

【００４３】

また、本実施例では、最初のメッセージであるか否かをメッセージ数カウンタの値に基づいて判定することにしたが、例えば、ＲＴＯＳにおける起床待ちキューに送信先のタスクのタスクＩＤがあるか否かによって判定するようにしてもよい。すなわち、図２のＳ１２０、Ｓ２３０の処理を行わず、図２のＳ１３０の処理に代えて、「ＲＴＯＳの起床待ちキューにタスクＢのタスクＩＤがないか」を判定し、ない場合（Ｓ１３０：ＹＥＳ）、Ｓ１４０の処理を行い、ある場合（Ｓ１４０：ＮＯ）、Ｓ１４０の処理を行わずに終了するようにする。また、Ｓ２４０の処理に代えて、タスクＢ用のキューにデータが残っていないかを判定し、残っている場合（Ｓ２４０：ＮＯ）Ｓ２１０へ移行し、残っていない場合（Ｓ２４０：ＮＯ）、データ受信利用処理を抜けるように構成すればよい。

【図面の簡単な説明】

【図１】実施例の処理実行装置としてのエンジン制御装置の構成を表すブロック図である。

【図２】実施例のメッセージ送信処理とデータ受信処理の流れを示すフローチャートである。

【図３】実施例のタスク間通信方法によって、タスクＡからタスクＢに対してメッセージを３回発行した場合の処理の様子を示す説明図である。

【図４】従来のメッセージ送信処理とデータ受信処理の流れを示すフローチャートである。

【図５】従来のタスク間通信方法によって、タスクＡからタスクＢに対してメッセージを３回発行した場合の処理の様子を示す説明図である。

【符号の説明】

１…ＥＣＵ

１０…マイコン

１１…ＣＰＵ

１２…ＲＯＭ

１３…ＲＡＭ

１４…Ｉ／Ｏ

２１…入力回路

２２…出力回路

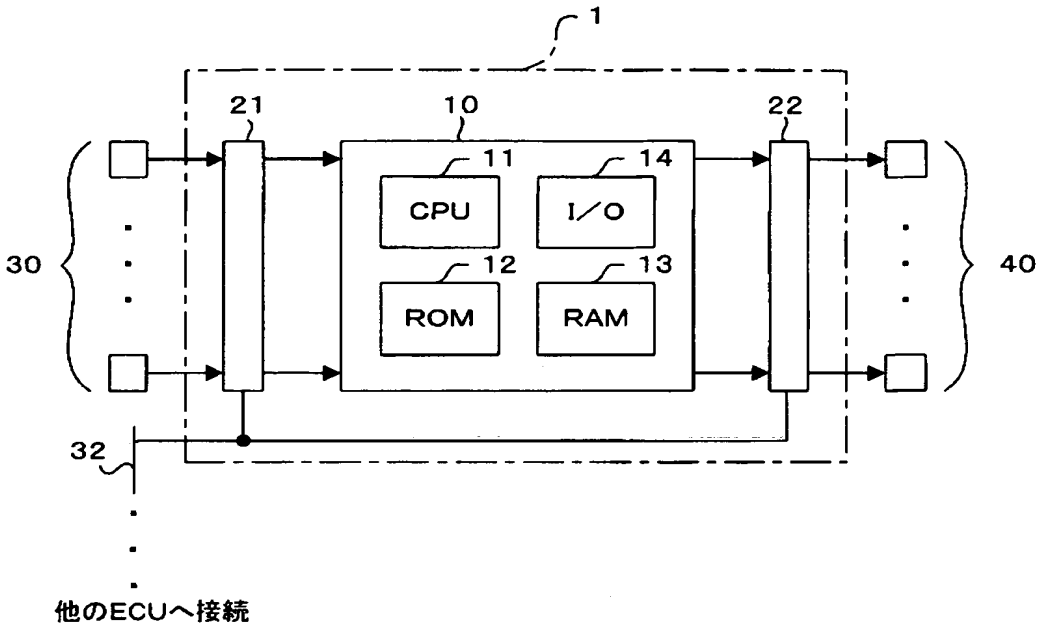
３０…センサ

３２…車内ＬＡＮ

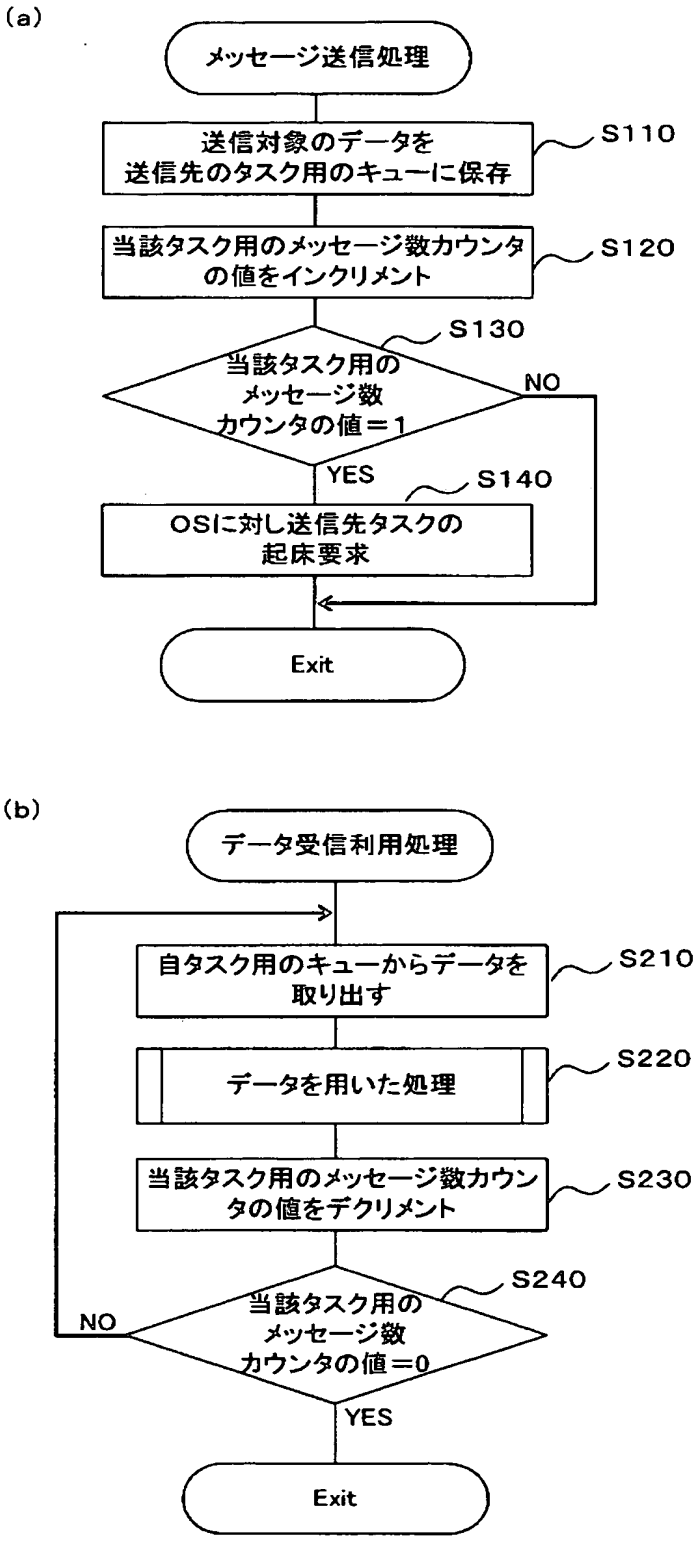
４０…アクチュエータ

【書類名】　　図面

【図１】

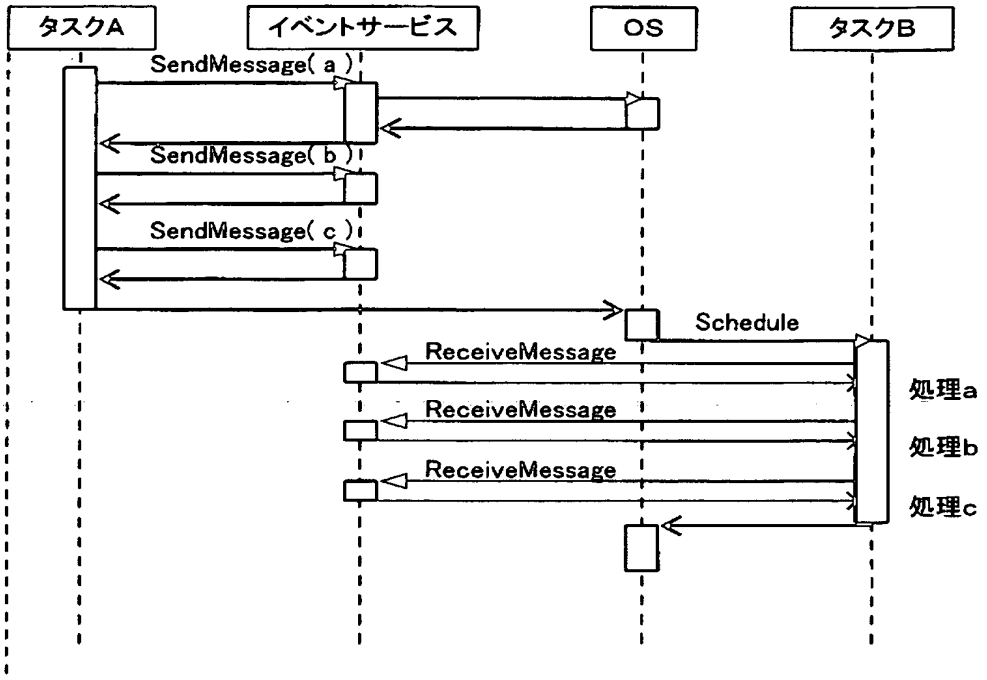


【図２】


(a)
メッセージ送信処理
送信対象のデータを
送信先のタスク用のキューに保存
S110
当該タスク用のメッセージ数カウンタ
の値をインクリメント
S120
S130
当該タスク用の
メッセージ数
カウンタの値＝1
NO
YES
S140
OSに対し送信先タスクの
起床要求
Exit
(b)
データ受信利用処理
自タスク用のキューからデータを
取り出す
S210
データを用いた処理
S220
当該タスク用のメッセージ数カウン
タの値をデクリメント
S230
S240
当該タスク用の
メッセージ数
カウンタの値＝0
NO
YES
Exit

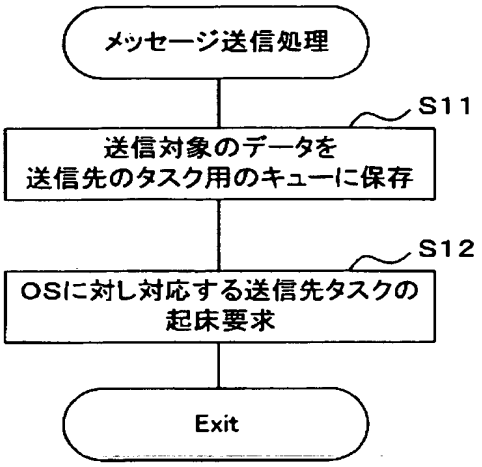
【図３】

タスクAからタスクBへのメッセージを3回連続して発行した場合の例

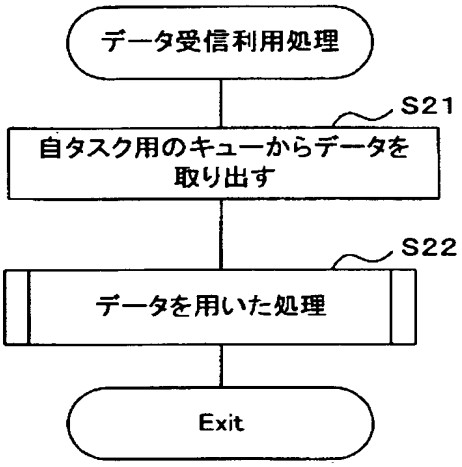


【図４】

(a)


メッセージ送信処理
S11
送信対象のデータを
送信先のタスク用のキューに保存
S12
OSに対し対応する送信先タスクの
起床要求
Exit

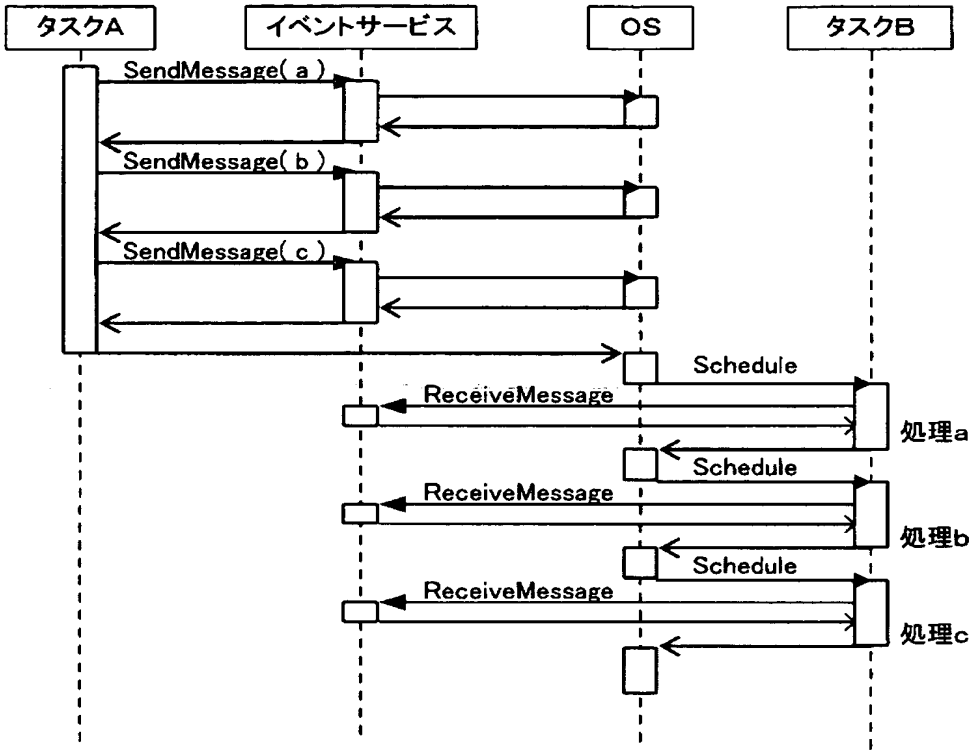
(b)


データ受信利用処理
S21
自タスク用のキューからデータを
取り出す
S22
データを用いた処理
Exit

【図５】

タスクAからタスクBへのメッセージを3回連続して発行した場合の例

・従来方式のシーケンス



【書類名】要約書

【要約】

【課題】処理装置の処理負荷を低減できるタスク間通信方法等を提供する。

【解決手段】メッセージ送信時にデータのキューイングを行った後、該当する優先度のメッセージ送信カウンタの値に基づき、現在送信待ちの同優先度のメッセージが有るか否かを確認する。待ちメッセージが無い場合は従来方式と同様にタスクＢの起床要求をＲＴＯＳに対して行うが、待ちメッセージがある場合はタスクＢの起床要求は行わず送信処理を終了する。そしてタスクＢが起床されると、キューイングしたデータを取り出してそのデータを利用する処理を行った後、さらに同優先度の送信待ちメッセージが残っているかどうかをチェックし、もし残っていればタスクを終了せずに続けて次の受信処理を実施し、送信待ちメッセージがなくなるまで繰り返す。このように構成することで、タスクの起床及び終了処理が少なく済み、処理負荷を低減できる。

【選択図】図３

特願２００３－０４００５６

出　願　人　履　歴　情　報

識別番号　　［０００００４２６０］

１．変更年月日　　１９９６年１０月　８日
［変更理由］　　名称変更
住　所　　愛知県刈谷市昭和町１丁目１番地
氏　名　　株式会社デンソー